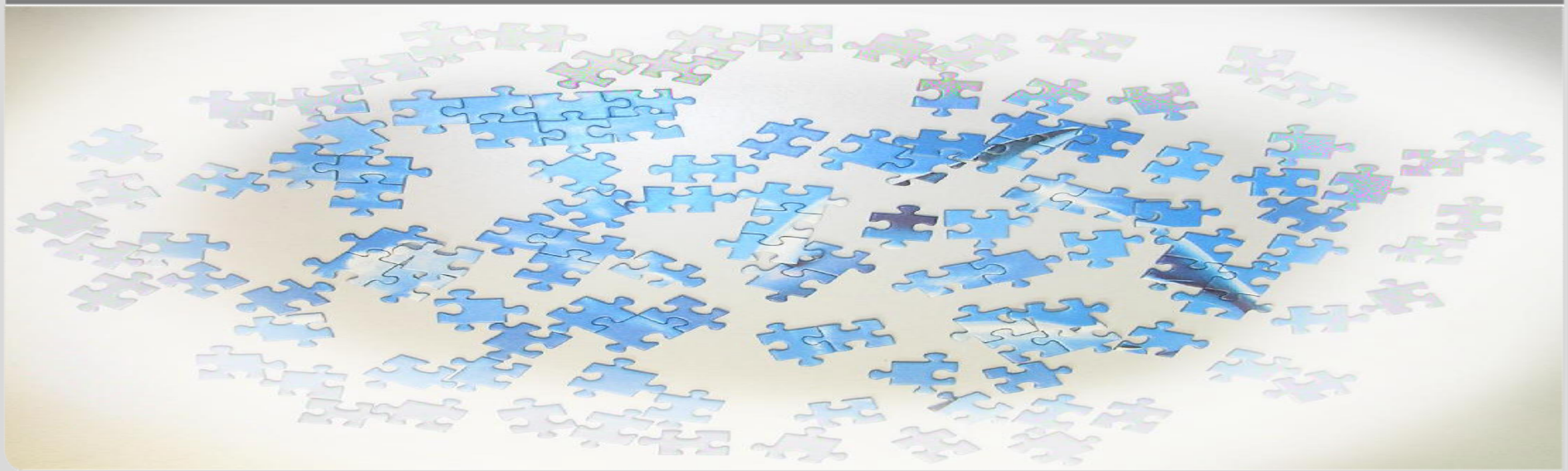
Karlsruhe Institute of Technology

METASONIC
process. in tune.

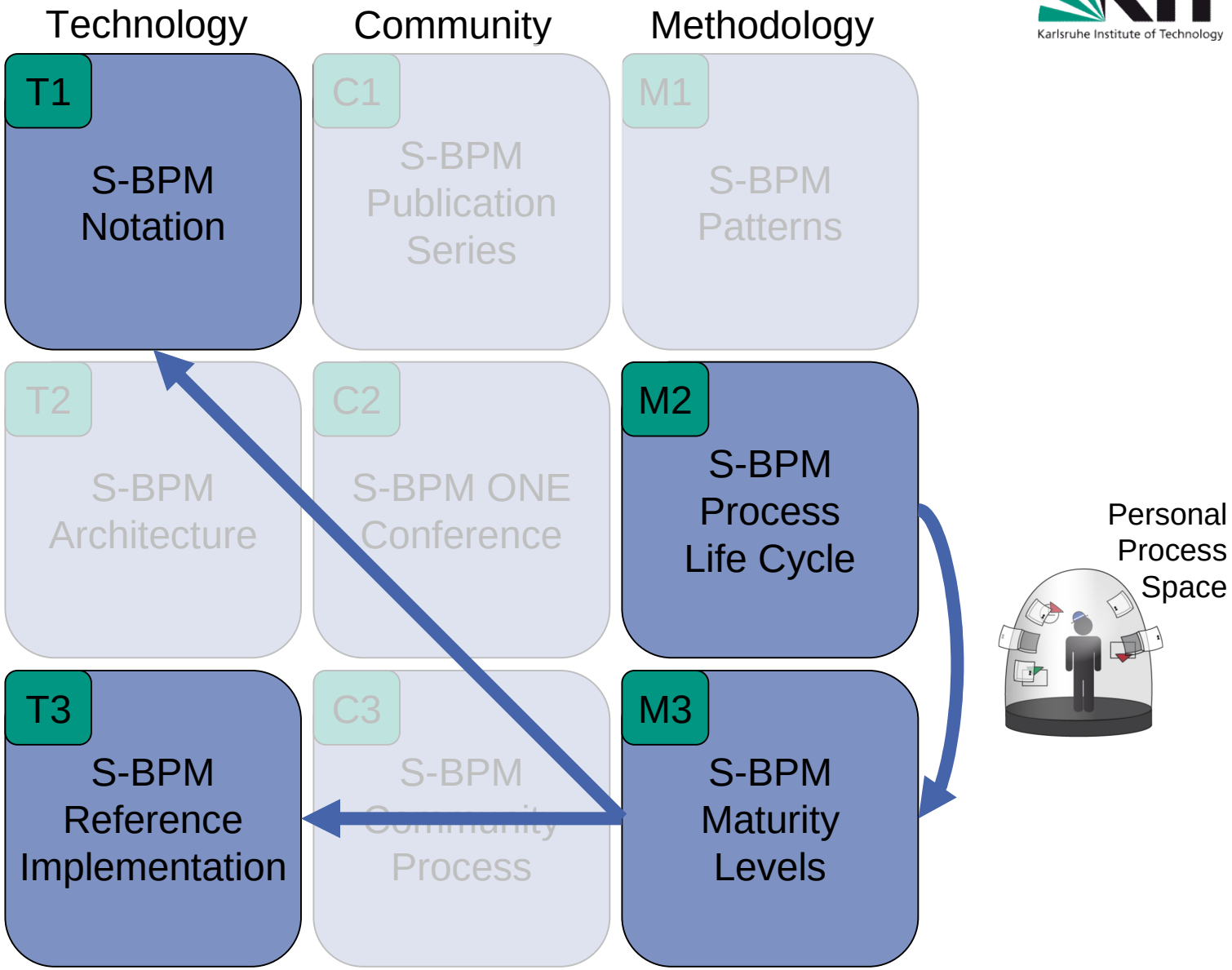
OOP and S-BPM – an analogy observation PowerSpeech

Karlsruhe, October 14th 2010
Hagen Buchwald, KIT, Institute AIFB

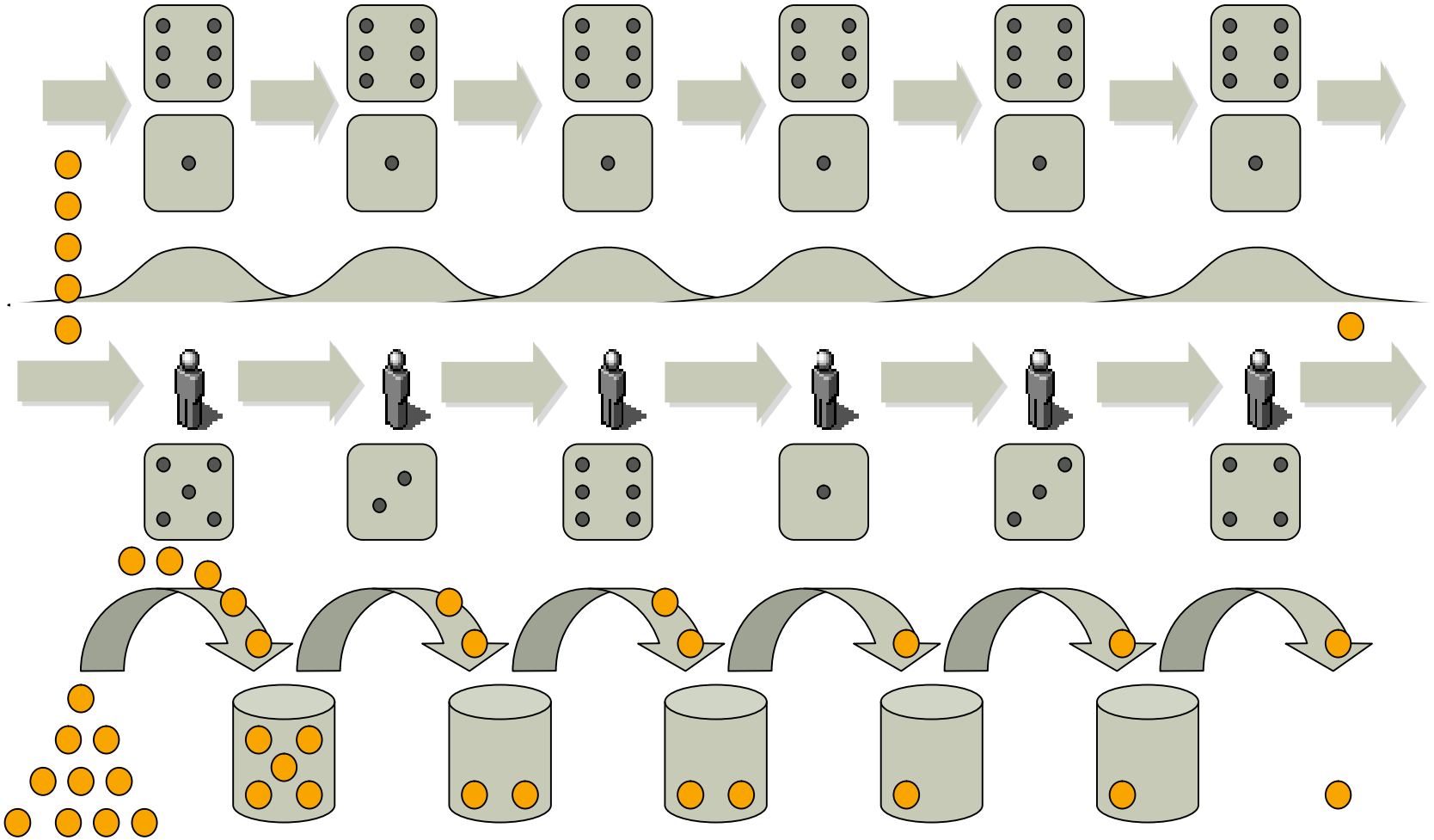
INSTITUTE AIFB – COMPLEXITY MANAGEMENT



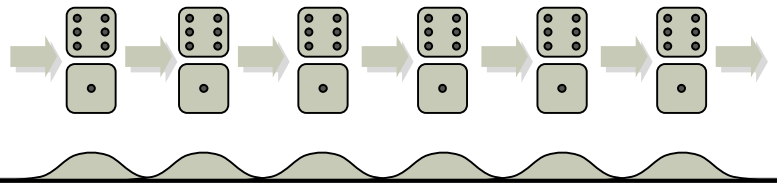
Potential Building Blocks of S-BPM (S-BPM ONE 2009)

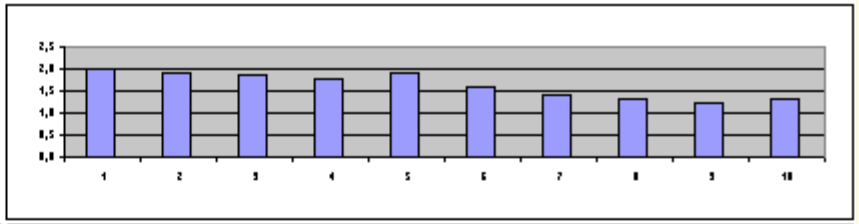


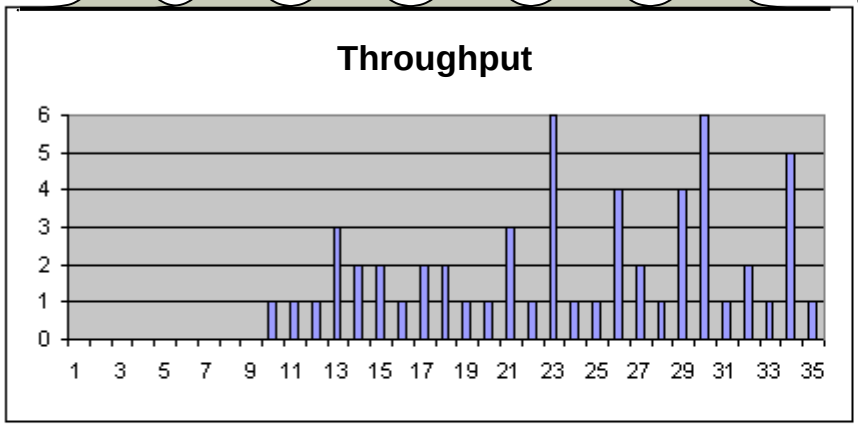
Why is a simple business process complex? The Dice Game.

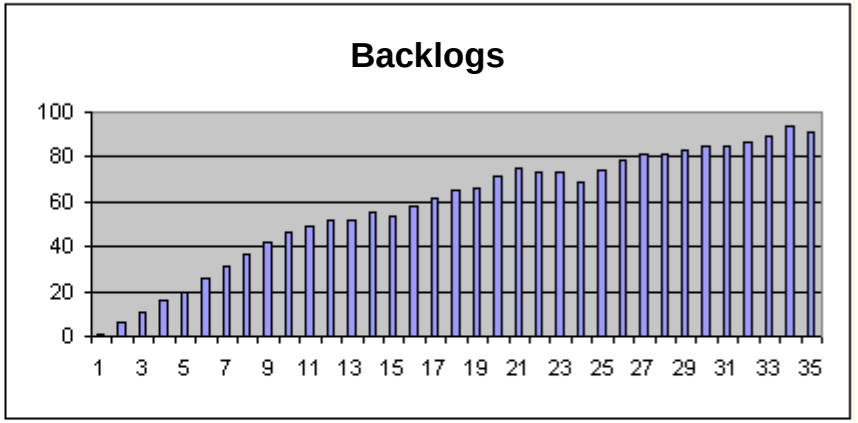


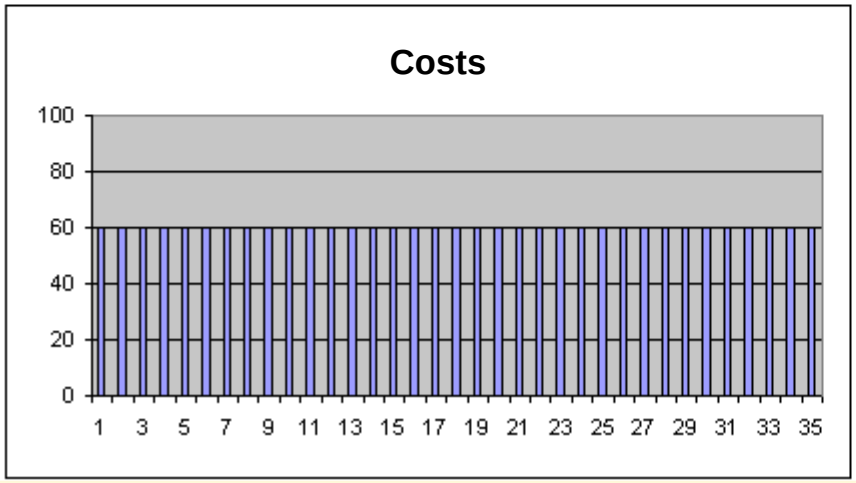
Why is a simple business process complex?

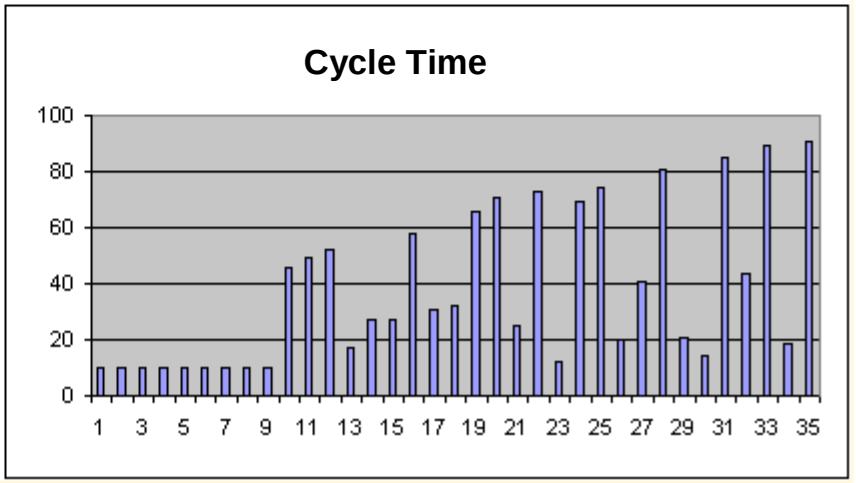




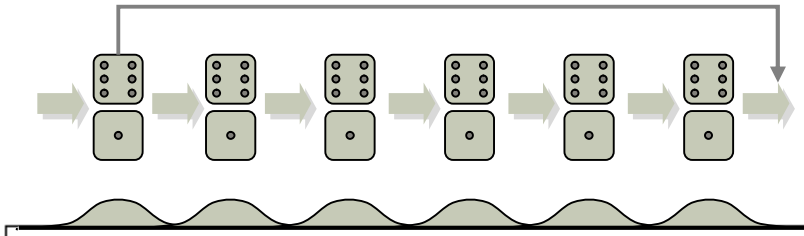


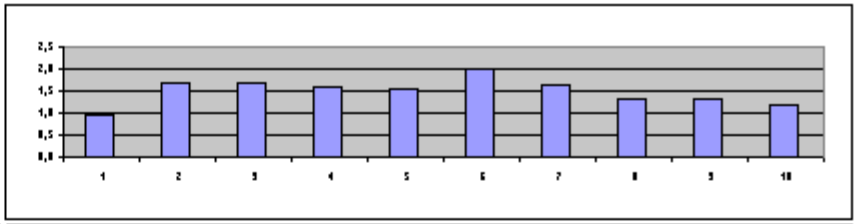


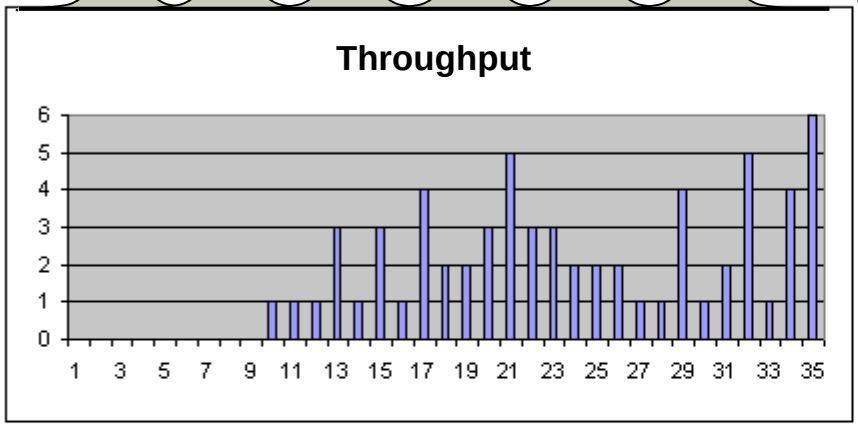


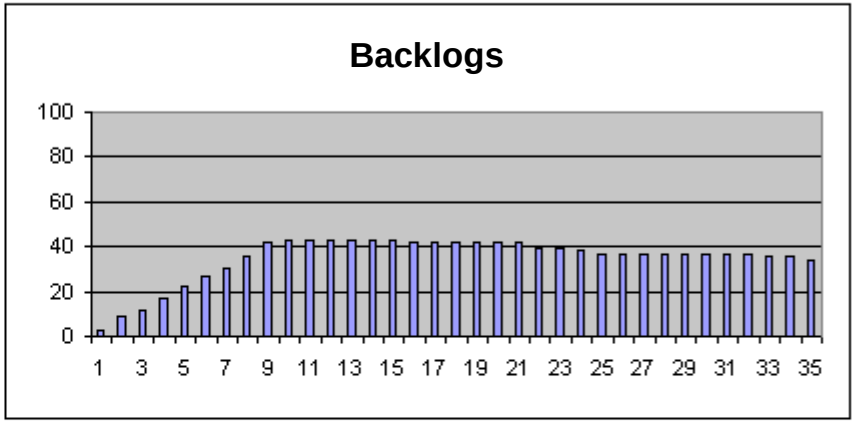


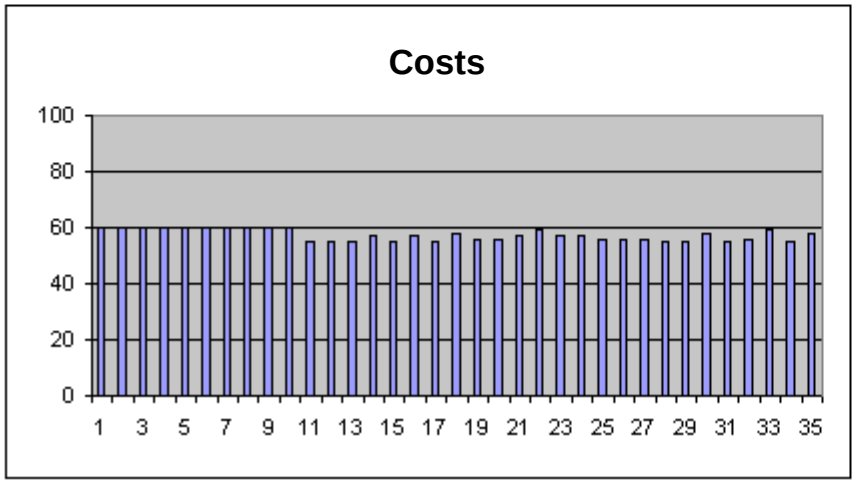
Why is a simple business process complex?

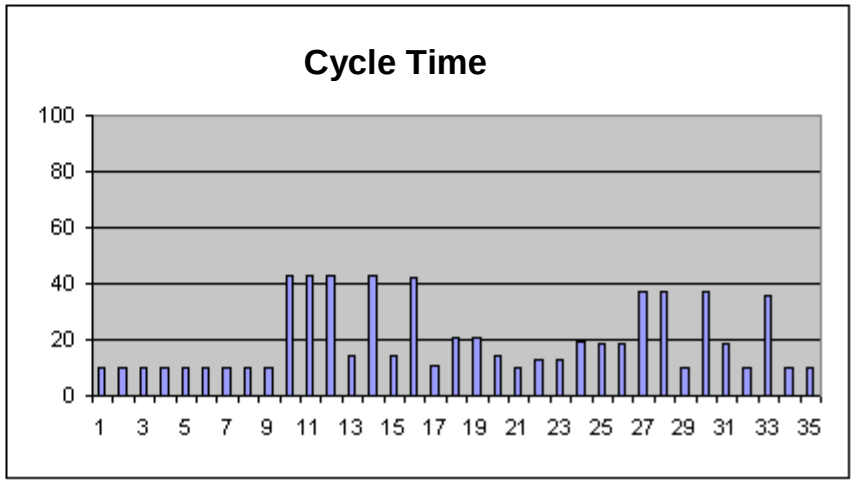




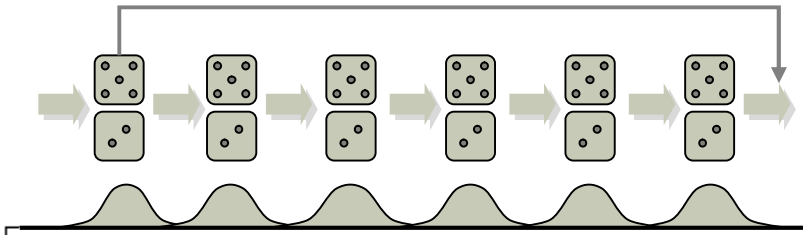


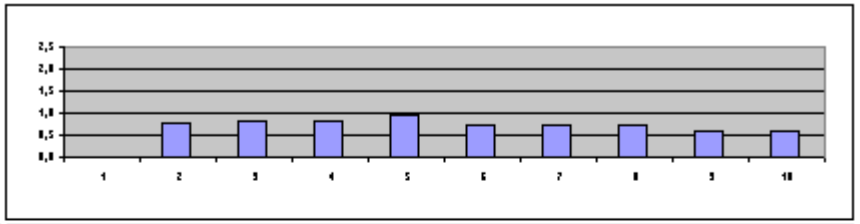


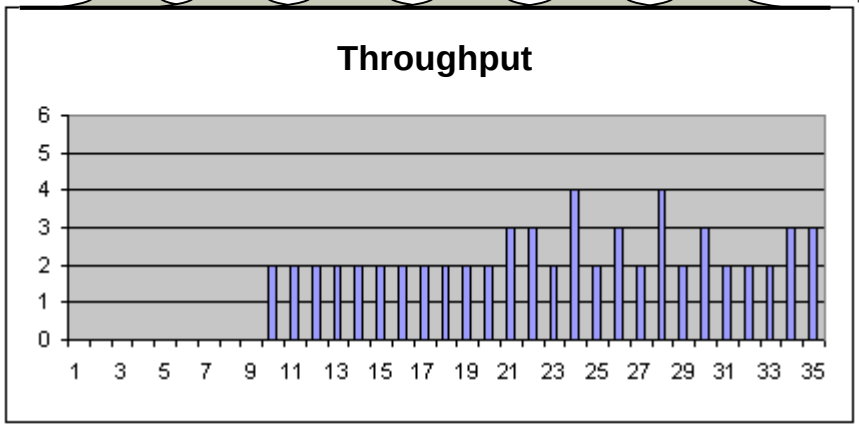


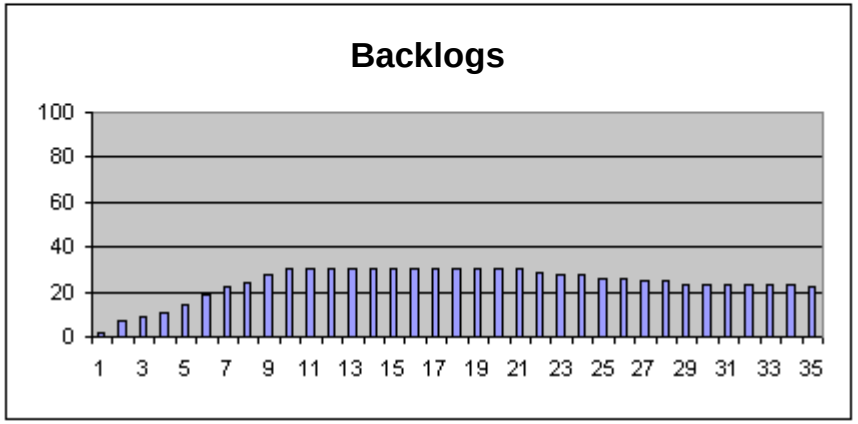


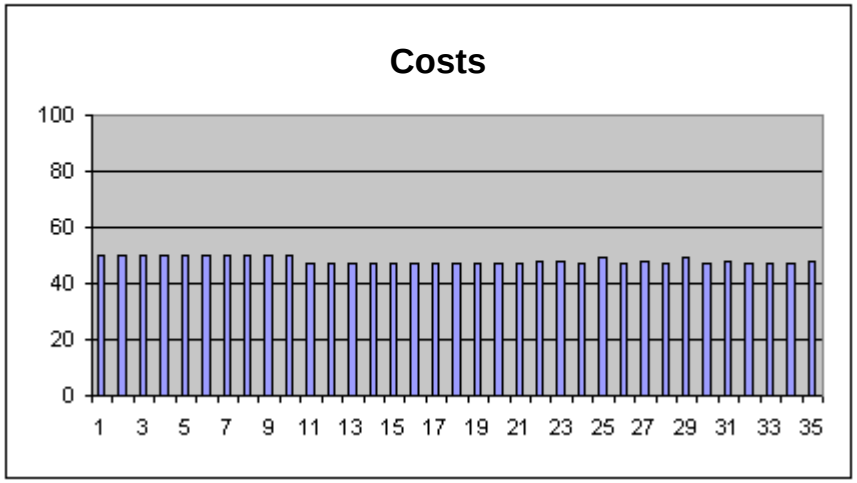
Why is a simple business process complex?

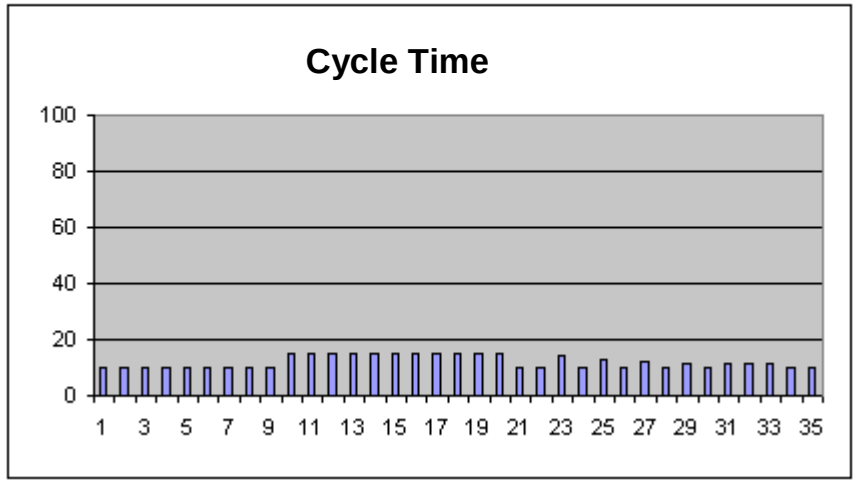




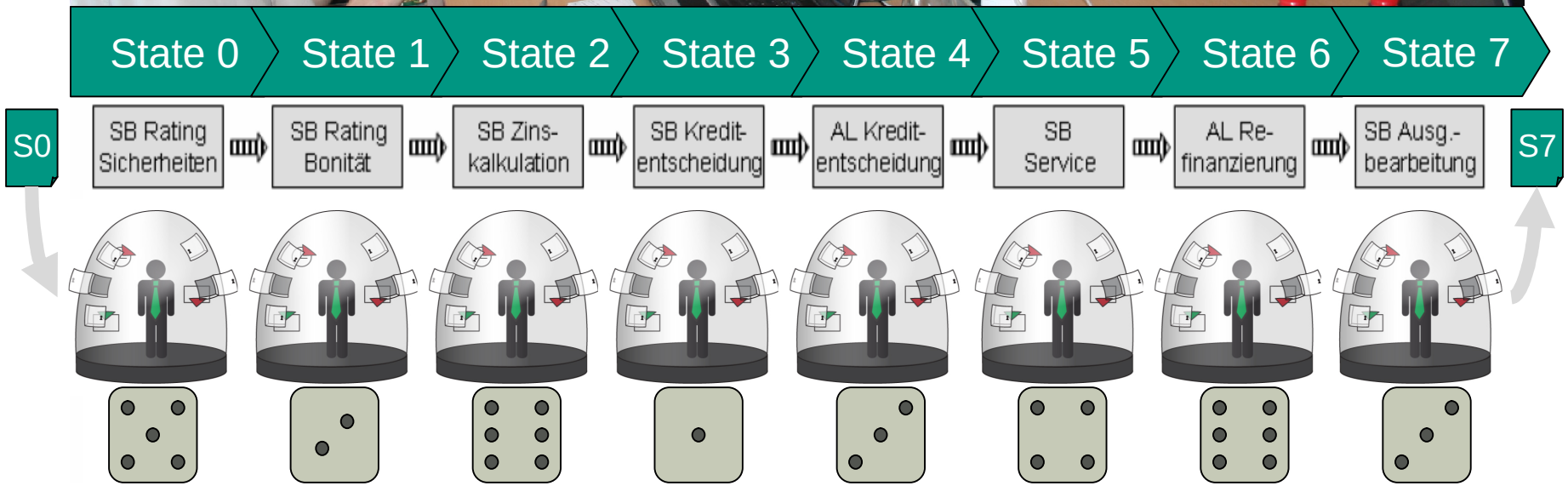






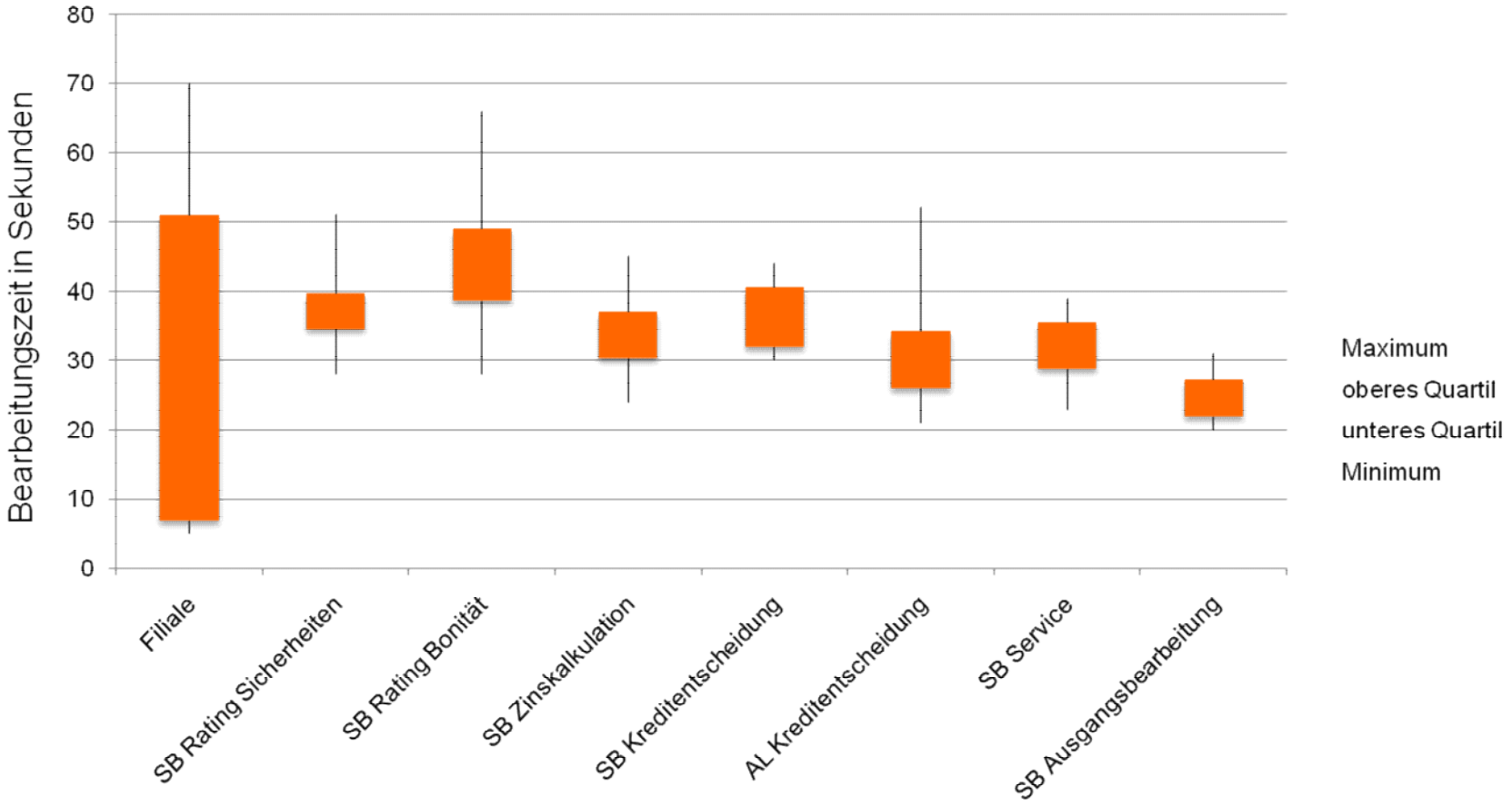


Why is a simple business process complex? The KreditSim role play.



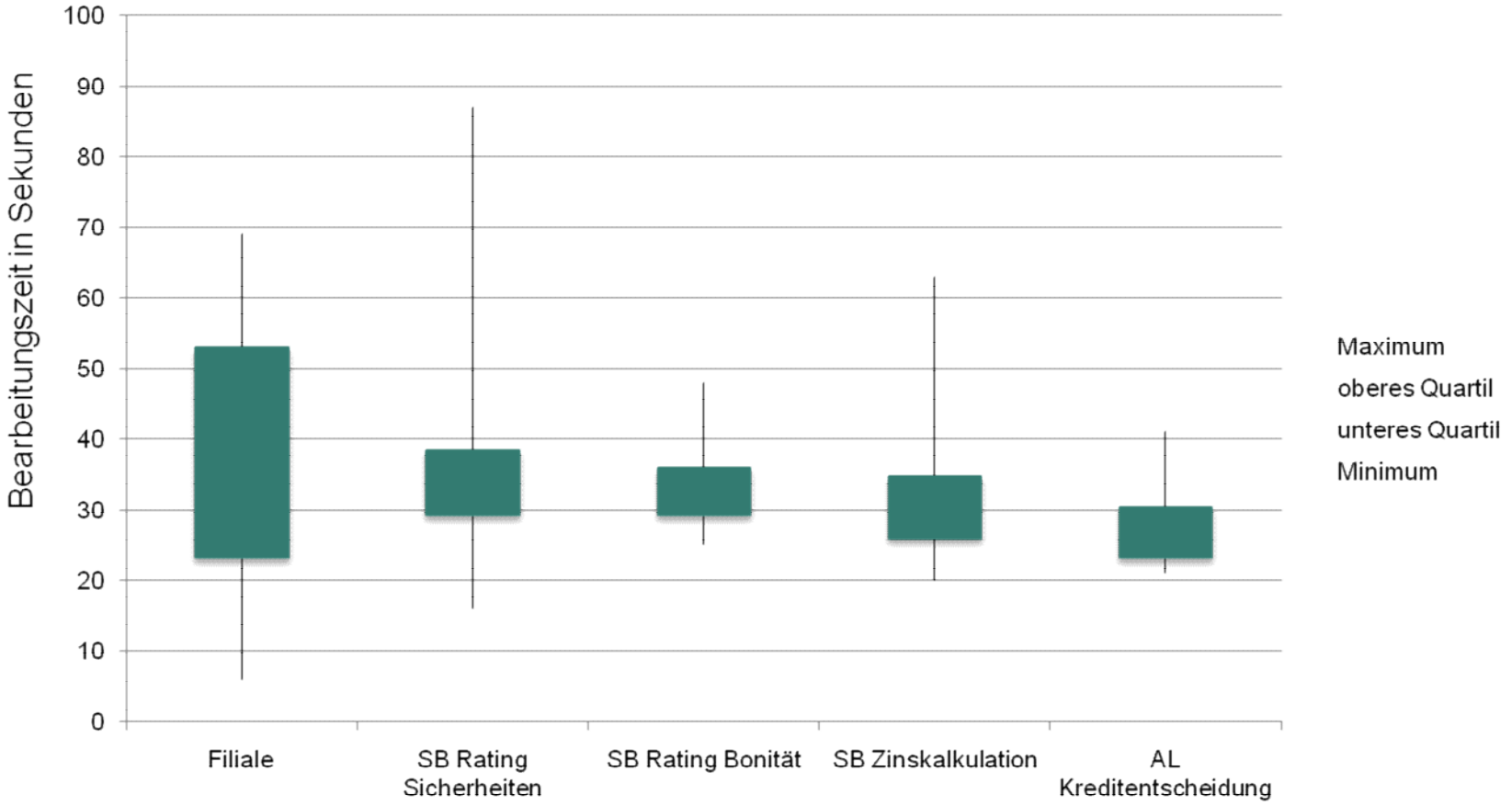
Why is a simple business process complex?
First answer: Process dynamics.

Variance

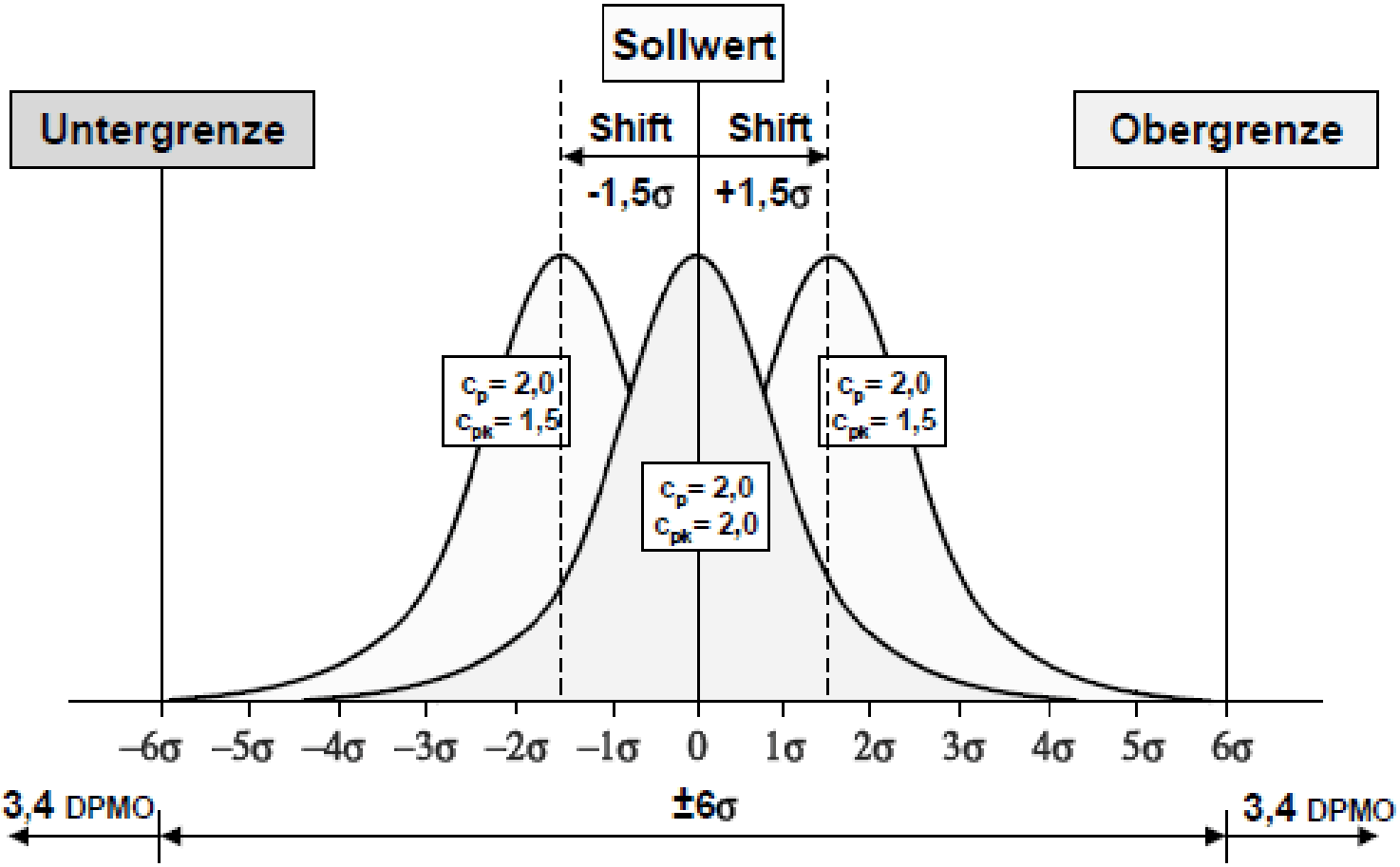


Why is a simple business process complex?
First answer: Process dynamics.

Variance

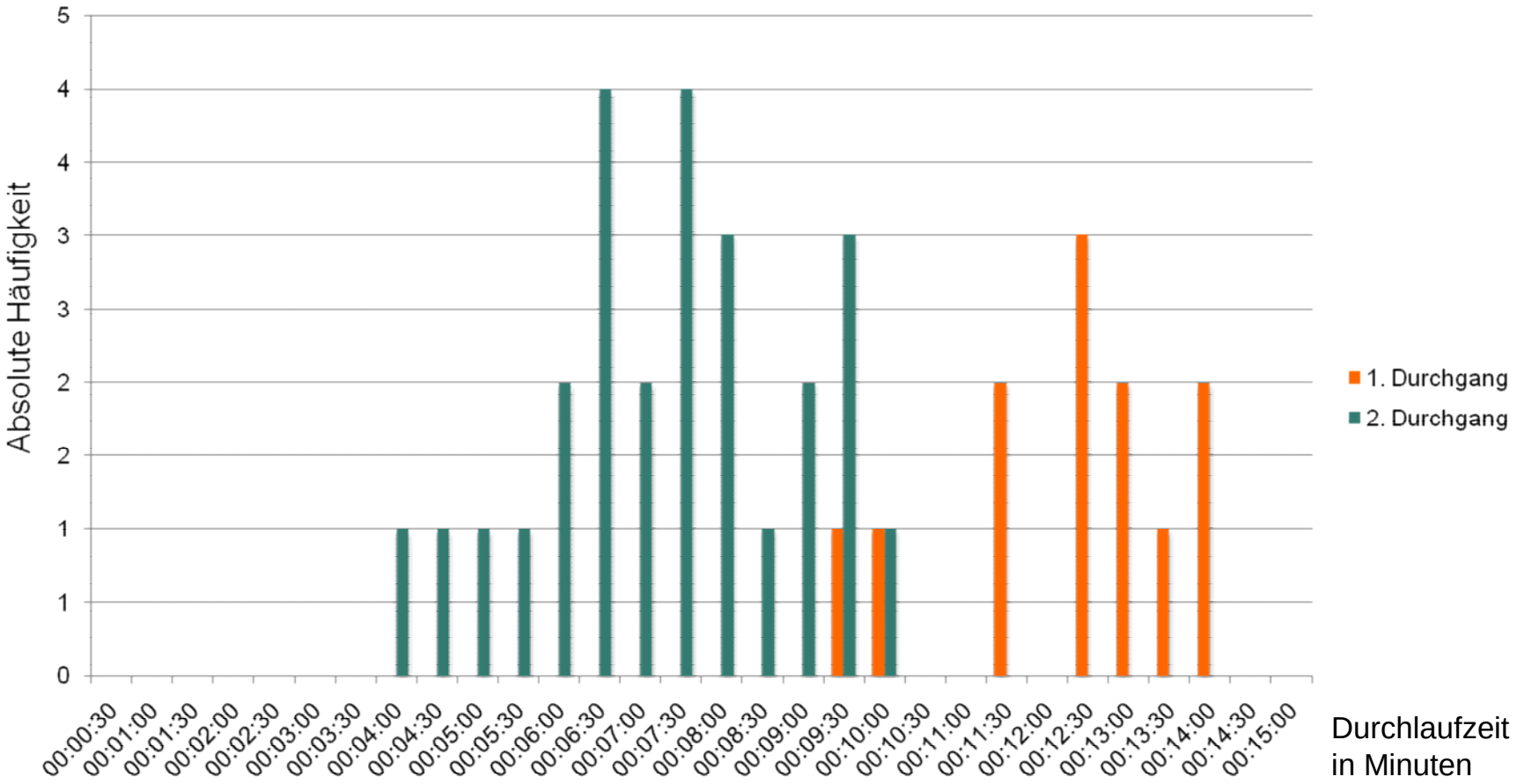


Why is a simple business process complex?
First answer: Process dynamics.

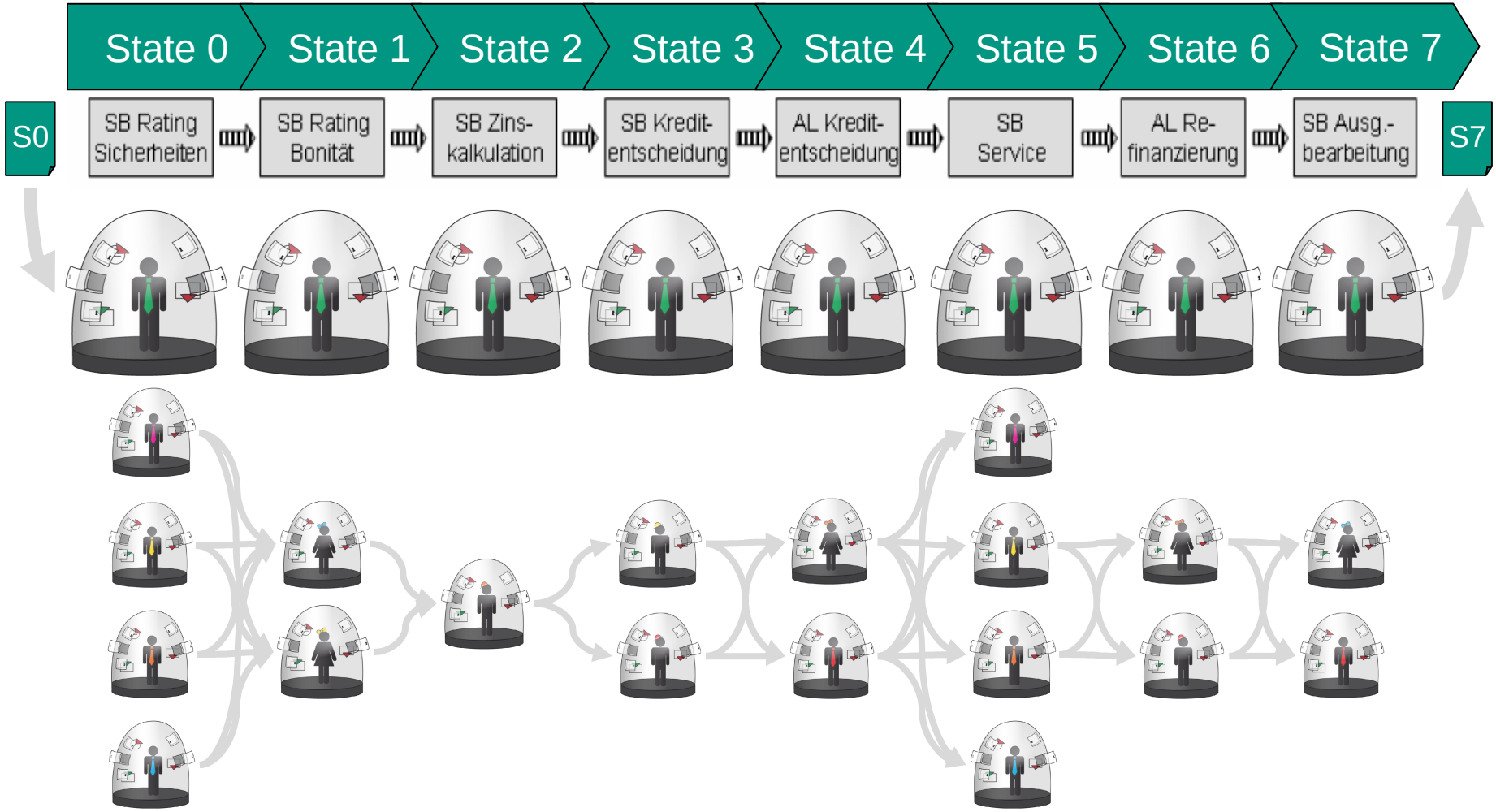


Why is a simple business process complex?
First answer: Process dynamics.

Histograms in comparison



For understanding the complexity of a business process, you first have to explore its natural context (nCEA).

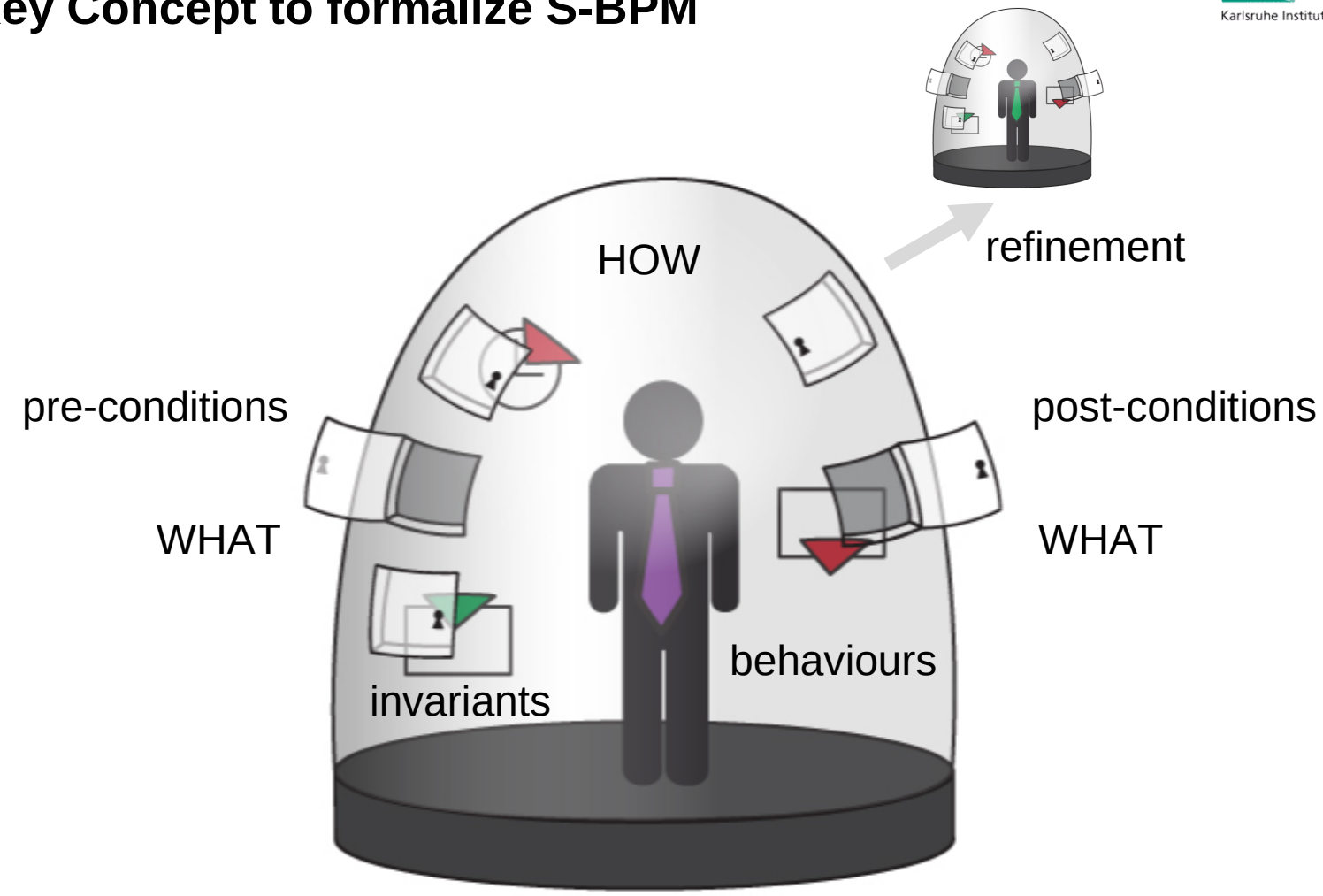


First Observation

1. The State is a Structure – a business object.
2. Reality is complex – there is a combinatorial number of paths from start to end.
3. We need a natural context exploration approach (nCEA, Matthes Elstermann, 2010) to understand (model – simulate – analyse) the complexity of a process.

ð How does that help us to find an adequate definition for S-BPM?

Personal Process Space
The Key Concept to formalize S-BPM



Excours: Abstract State Machines
Yuri Gurevich, mid 1980

- An Abstract State Machine (ASM) is a state machine operating on states which are arbitrary data structures.
- A state transition can only be executed if the pre-conditions for that transition are fulfilled.
- The state transitions involve only a bounded part of the state, described by post-conditions
- and everything is invariant under isomorphisms of structures.
- Pre- and post-conditions describe the WHAT-part of a transition – the specification.
- The How-part – the implementation – is described by the behaviour of the data structure, which acts like an algebra, always guaranteeing the consistency of the invariant.
- An Abstract Data Type (ADT) is an algebra describing the WHAT-part of an abitrary data structure. It is the core concept of OOP.
- Thesis #1: ASMs are an excellent candidate as core concept of S-BPM.
- Thesis #2: The states of these ASMs must be described as ADTs.

OOP Maturity Levels

Karlsruhe Institute of Technology

Level 7
Multiple inheritance
You may declare classes which inherit from more than one parent class.

Level 6
Polymorphism and dynamic binding
Elements of a system may reference objects of more than one class,
and routines may have different implementations in different classes.

Level 5
Inheritance
A class can be defined as an reduction or extension of another class.

Level 4
Classes
Classes are implementations of abstract data types (ADTs).

Level 3
Garbage collection
Unreferenced objects should be deallocated automatically
by the underlying runtime system.

Level 2
Data abstraction
Objects must be described as implementations of abstract data types (ADTs).

Level 1
Objectbased, modular structure
Systems a modularized based on their data structures.

Source: Bertrand Meyer, Object-Oriented Software Construction, 1986

S-BPM Maturity Levels

Level 7
Multiple inheritance
You may declare classes which inherit from more than one parent class.

Level 6
Polymorphism and dynamic binding
Elements of a system may reference subjects of more than one class,
and behaviour may have different implementations in different classes.

Level 5
Inheritance (Refinement)
A class can be defined as an reduction or extension of another class.

Level 4
Classes
Classes are implementations of abstract state machines (ASMs).

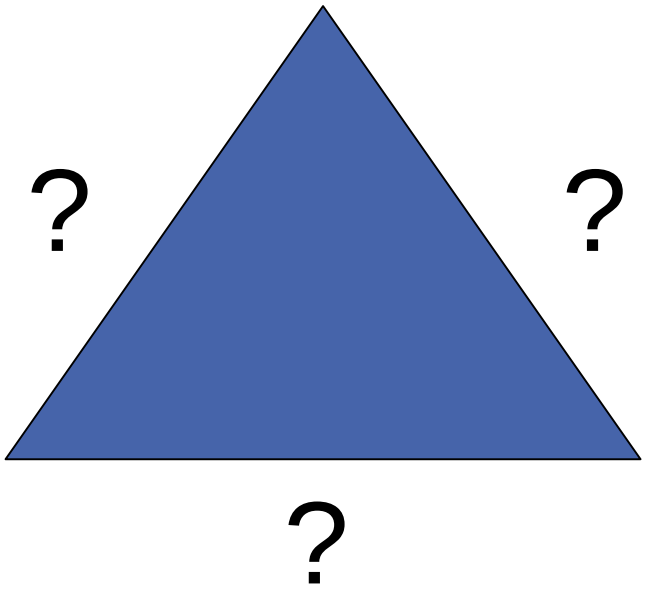
Level 3
Linear scalability
The more cores the underlying hardware offers,
the more subjects can act in parallel in order to respond to higher market demand.

Level 2
Behaviour abstraction
Subjects must be described as implementations of abstract state machines (ASMs).

Level 1
Subjectbased, modular structure
Systems a modularized based on their subject‘s behaviour (habits).

Source: Bertrand Meyer, Object-Oriented Software Construction, 1986

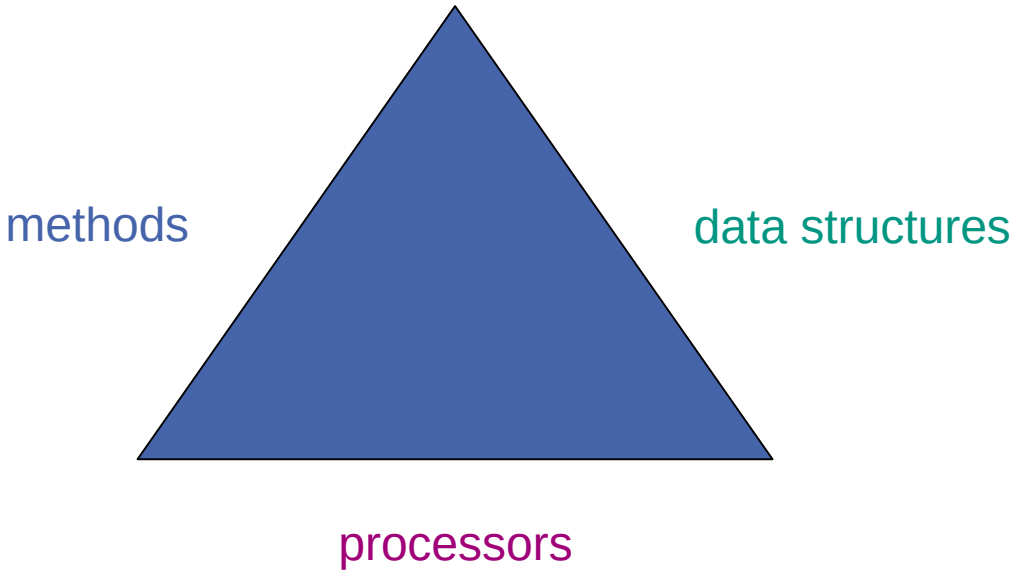
S-BPM-Systems are Software Systems. But what is a software system?



Source: Bertrand Meyer, Object-Oriented Software Construction, 1997

S-BPM-Systems are Software Systems.
But what is a software system?

A software system consists of processors,
which execute methods on data structures.



The difference between OOP and S-BPM is our primary question to the system:

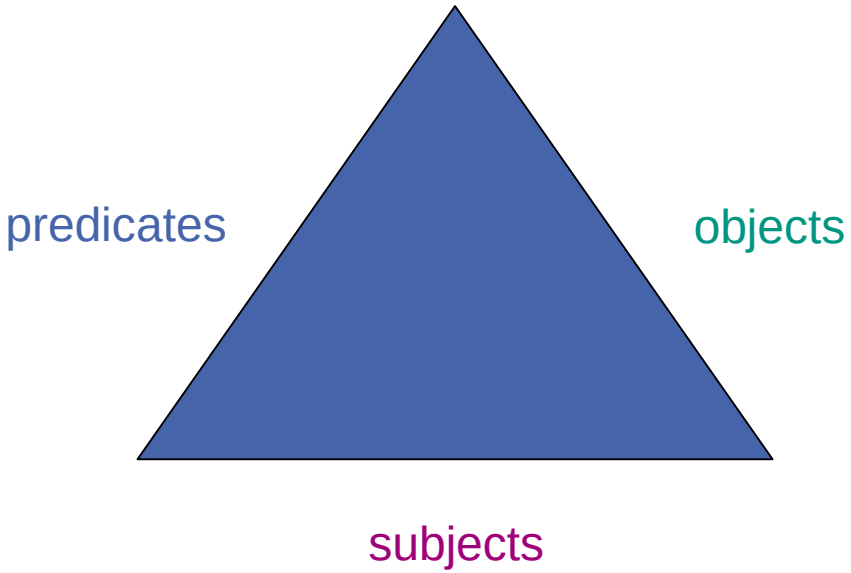
OOP: On what does the system work?

S-BPM: Who acts in the system?

Source: Bertrand Meyer, Object-Oriented Software Construction, 1997

S-BPM-Systems are Software Systems.
But what is a software system?

A S-BPM system consists of subjects,
which execute predicates on objects.



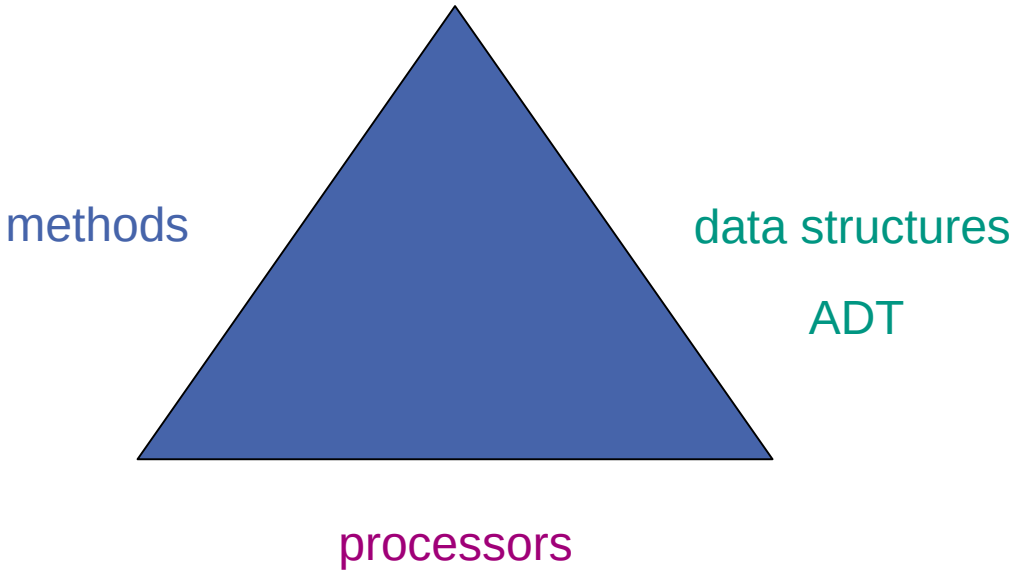
The difference between OOP and S-BPM is our primary question to the system:

OOP: On what does the system work?

S-BPM: Who acts in the system?

What is OOP?
A formal definition.

A software system consists of processors,
which execute methods on data structures.

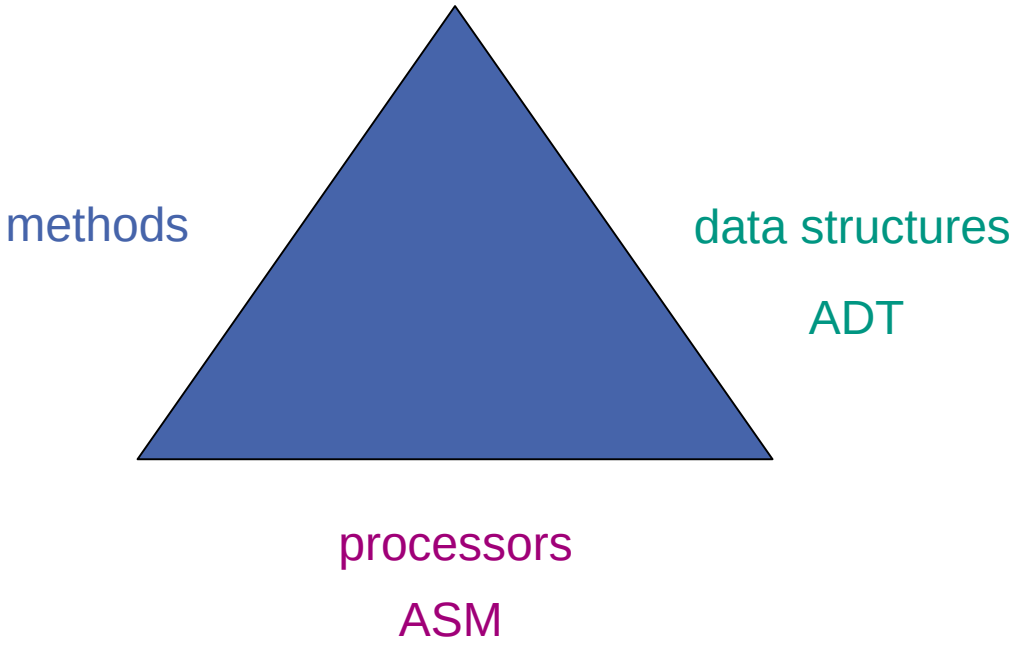


OOP is the development of software systems as a structured collection of implementations of Abstract Data Types (ADTs).

Source: Bertrand Meyer, Object-Oriented Software Construction, 1997

What is S-SPM?
A proposal of a formal definition.

A software system consists of processors,
which execute methods on data structures.



S-BPM is the development of software systems as a structured collection of implementations of multi-agent Abstract State Machines (ASMs) with ADTs as states.

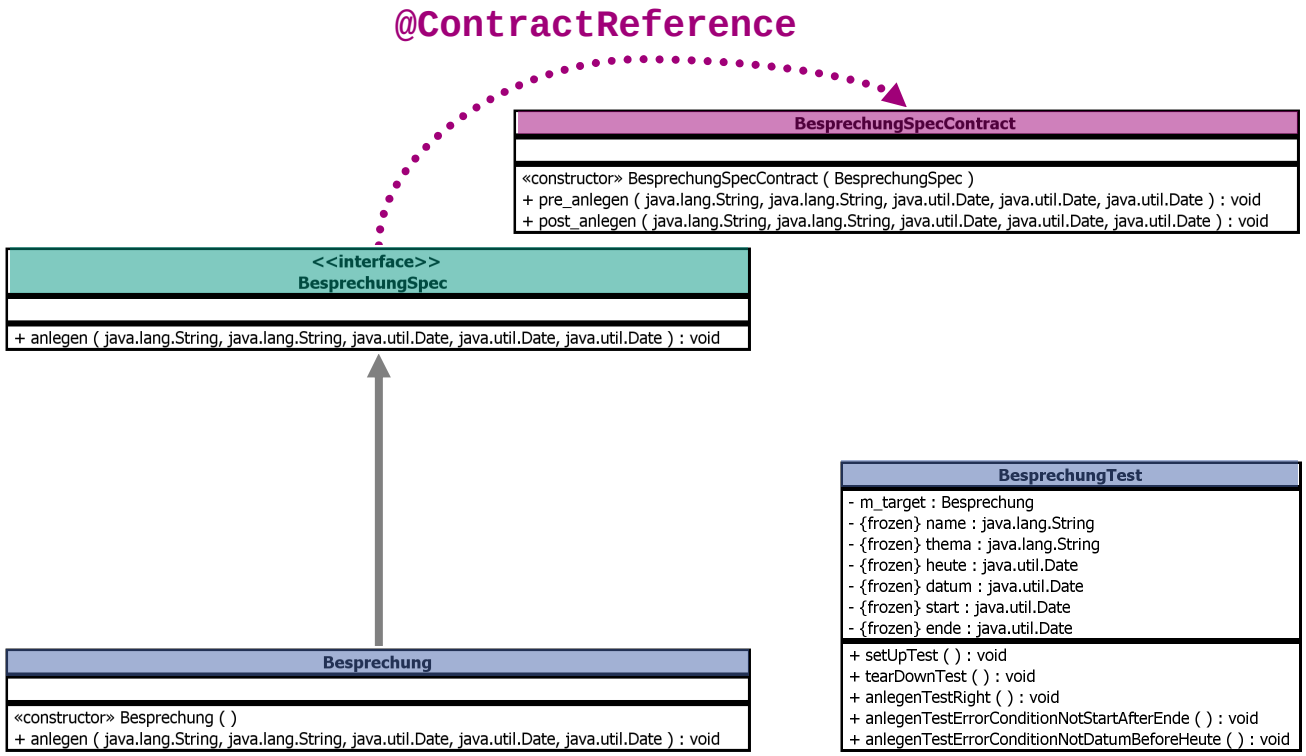
Each operative class is completely described by its `Interface` and the corresponding `ContractClass`.

Specification Level
ð Abstract Data Type

WHAT

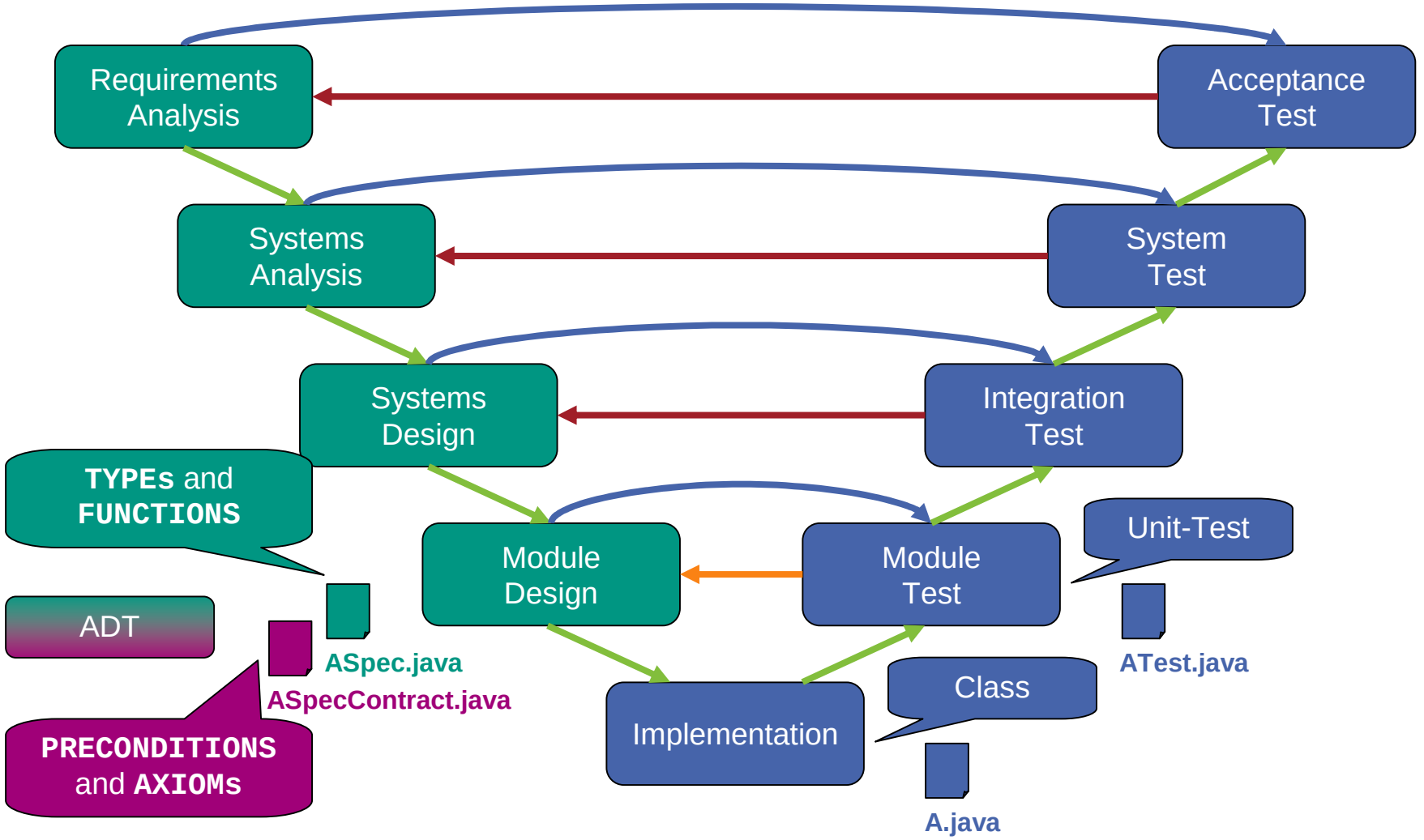
Implementation Level

HOW

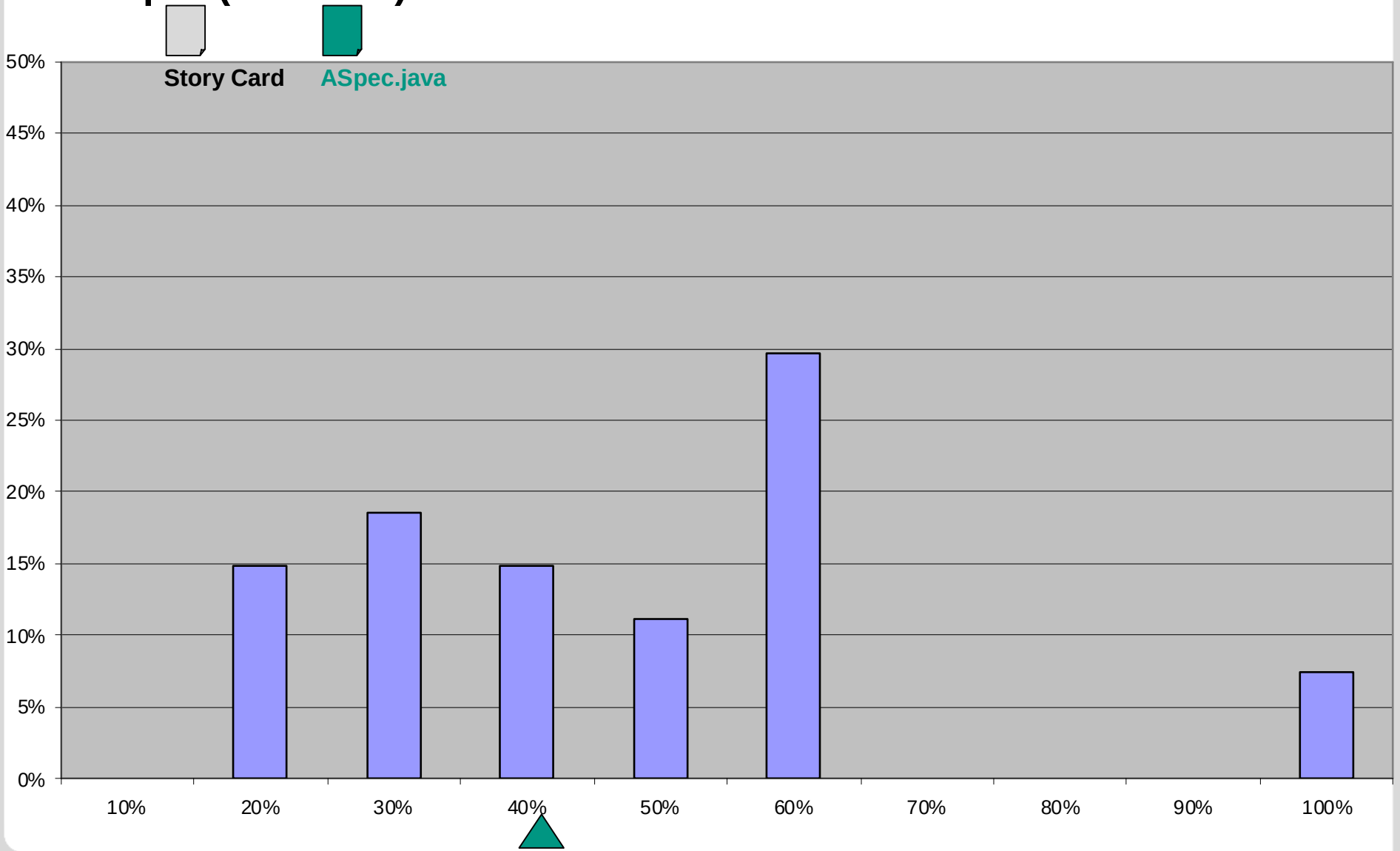


Objectoriented Programming is the development of software systems as structured collections of Implementations of abstract data types (ADT).

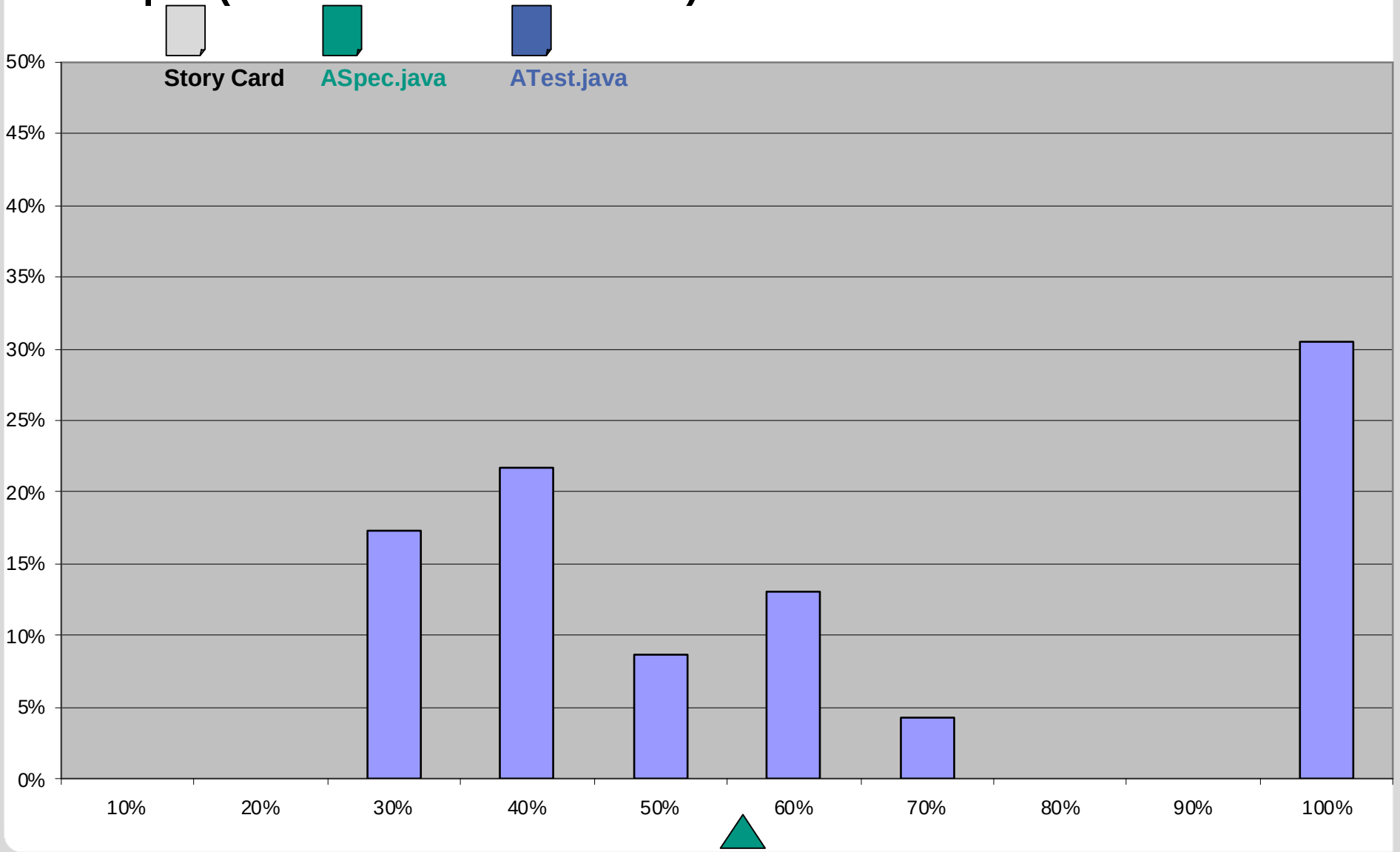
Interfaces specify the syntax part, Contract-Classes the semantic part of the ADT.



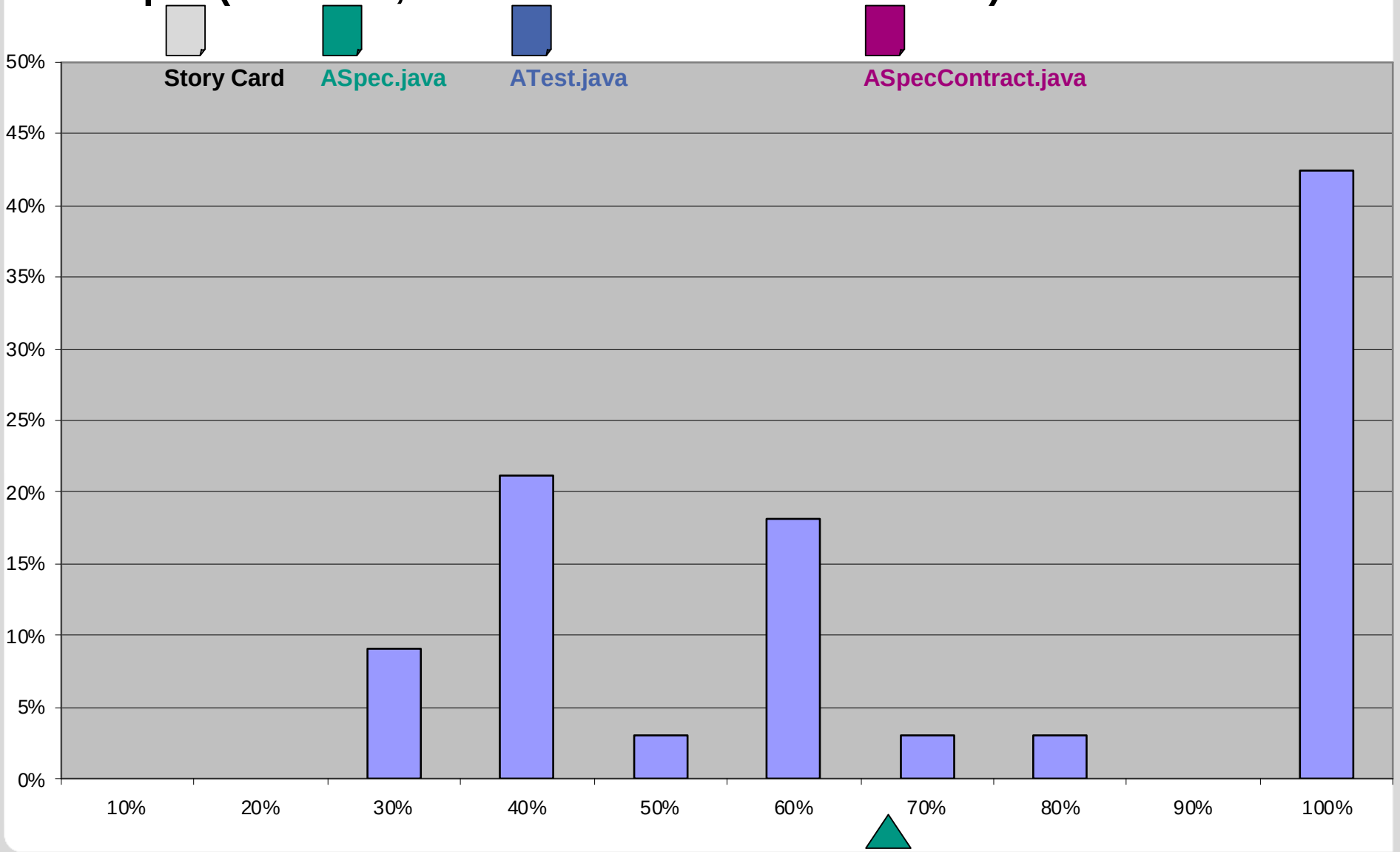
Results of our Design by Contract experiments SS2010 Group A (Interface)



Results of our Design by Contract experiments SS2010 Group B (Interface and Test-Class)

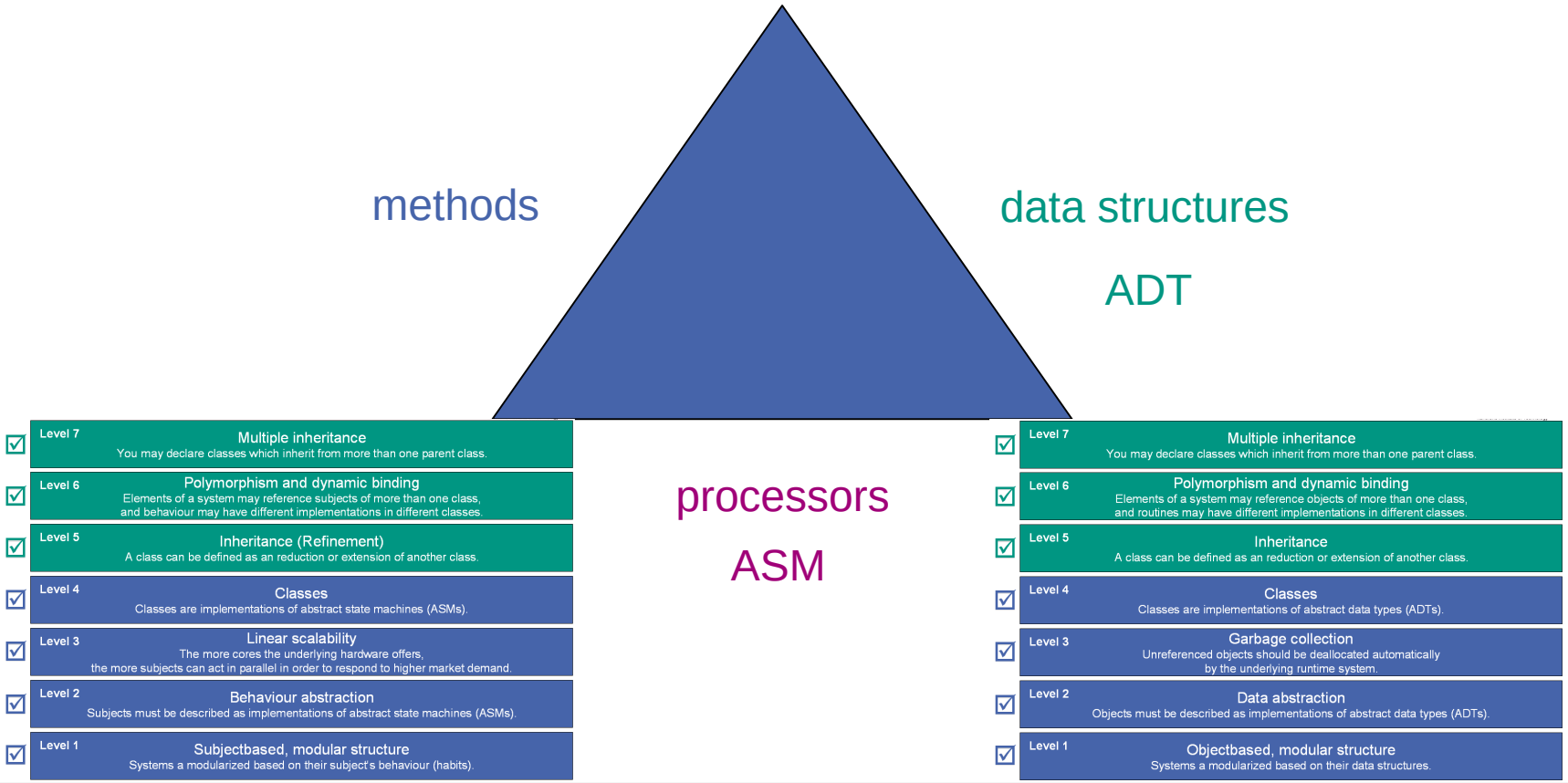


Results of our Design by Contract experiments SS2010 Group C (Interface, Test-Class and Contract-Class)



What is S-SPM?
A proposal of a formal definition.

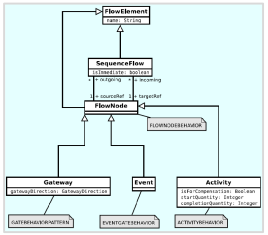
A software system consists of processors,
which execute methods on data structures.



Conclusion

Technology | Community | Methodology

Comprehensive ASM Specification of PASS



Egon Börger

T1 S-BPM Notation

C1 S-BPM Publication Series

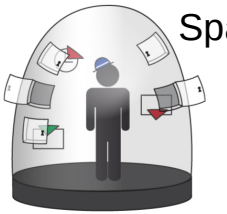
M1 S-BPM Patterns

T2 S-BPM Architecture

C2 S-BPM ONE Conference

M2 S-BPM Process Life Cycle

Personal Process Space



T3 S-BPM Reference Implementation

C3 S-BPM Community Process

M3 S-BPM Maturity Levels